PATENT
Attorney Docket No. GB919990026US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

Henry Esmond Butterworth

Serial No: 09/338,035

Filed: June 22, 1999

For: DATA PROCESSING SYSTEMS AND METHOD FOR PROCESSING TASKS IN SUCH SYSTEMS

Examiner: Kenneth TANG

Art Unit: 2127

CERTIFICATE OF SUBMISSION BY FACSIMILE

PTO FAX NUMBER: 571-273-8300
TOTAL NUMBER OF PAGES: 8

Dear Sir:

I hereby certify that the following documents are being transmitted to the U.S. Patent and Trademark Office on the date shown below:

1. NOTICE OF APPEAL (1 page);
2. PRE-APPEAL BRIEF REQUEST FOR REVIEW (1 page);
3. REASONS FOR REQUEST FOR REVIEW (5 pages); and
4. this CERTIFICATE OF SUBMISSION BY FACSIMILE (1 page).

If you did not receive all the pages, please telephone us at 718-544-1110, or fax us at 718-544-8588.

Respectfully submitted,

Dated: July 21, 2005

Ido Tuchman, Reg. No. 45,924
Law Office of Ido Tuchman
69-60 108th Street, Suite 503
Forest Hills, NY 11375
Telephone (718) 544-1110
Facsimile (718) 544-8588

PATENT
Attorney Docket No. GB919990026US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	
Henry Esmond Butterworth	Examiner: Kenneth TANG
Serial No: 09/338,035	Art Unit: 2127
Filed: June 22, 1999	
For: DATA PROCESSING SYSTEMS AND METHOD FOR PROCESSING TASKS IN SUCH SYSTEMS	

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a notice of appeal.

The review is requested for the reasons stated on the attached sheets.

Respectfully submitted,

Dated: July 21, 2005

Ido Tuchman, Reg. No. 45,924
Law Office of Ido Tuchman
69-60 108th Street, Suite 503
Forest Hills, NY 11375
Telephone (718) 544-1110
Facsimile (718) 544-8588

REASONS FOR REQUEST FOR REVIEW

Claims 11 and 15

Independent claims 11 and 15 of the immediate Application stand rejected as obvious over U.S. Patent No. 5,745,778 to Alfieri ("Alfieri") in view of U.S. Patent No. 5,778,434 to Nguyen et al. ("Nguyen"). Final Office Action, paragraph 4. The Examiner applies similar reasoning to reject claims 11 and 15. Applicant therefore uses claim 11 as a representative claim to discuss both claims.

In rejecting claim 11, the Office Action alleges that Alfieri teaches placing tasks of the same task type into a batch. Final Office Action, paragraph 5. Assuming for the moment that the Examiner's interpretation of Alfieri is correct, the Applicant respectfully submits claim 11 does not recite batching tasks "of the same type."

Claim 11 recites, "batching the new task with the cached task if the task queue includes the cached task that requires the same code to process the cached task and the new task."

The Examiner states, "Alfieri fails to explicitly teach that the caching consists when it has the same code (instead of 'like type')" but that memory caching is well known and expected in the art. Final Office Action, paragraph 12. The Applicant respectfully submits that memory caching operations are wholly different from task batching operations. Claim 11 recites "batching the new task with the cached task . . . ."

Along similar lines, Office Action takes Official Notice that "memory caching of the same code or instructions is well known and expected in the art." Final Office Action, paragraph 12. The Applicant finds no relevance in this assertion since memory caching is the process by which instructions or data are copied from slow-access memory to fast-access memory, and is wholly different from task batching operations. Furthermore, the Applicant does not understand what the Examiner means by "memory caching of the same code or instructions" and respectfully requests documentary evidence supporting such a statement in accordance with MPEP 2144.03.

The Office Action additionally states that the broadest reasonable interpretation of a thread can be generalized as a computer instruction.

Final Office Action, paragraph 12. The relevance of this assertion is lost on the Applicant. The Examiner equates threads with computer instructions, but does not provide evidence in the record that Alfieri teaches "batching the new task with the cached task if the task queue includes . . . same code to process the cached task and the new task," as recited in claim 11.

The above discussion assumes that the Examiner's assertion that Alfieri teaches placing tasks "of the same type" into a batch is correct. The Applicant, however, respectfully submits that the Examiner's interpretation of Alfieri is incorrect. According to Alfieri, a "thread group" is a set of closely related threads within a process that tend to access and operate on the same data. Alfieri, col. 2, lines 61-63. Thus, Alfieri appears to teach grouping threads according to the data they process, rather than grouping tasks "of the same type," as alleged in the Office Action.

Finally, the Office Action states, "Nguyen teaches a queue (batch) that hold the same identifier that processes tasks out of sequential order." Final Office Action, paragraph 6. This statement implies the Examiner equates queuing operations with batching operations. It is respectfully submitted that the Office Action appears to improperly equate queuing operations with batching operations. Furthermore, the Applicant respectfully submits that Nguyen never mentions batching operations and no evidence has been presented by the Examiner showing such a teaching.

For at least these reasons, it is respectfully submitted that the rejections of claims 11 and 15 are inappropriate and should be withdrawn.

Claims 1, 8 and 9

Independent claims 1, 8 and 9 stand rejected as obvious over Alfieri and Nguyen. Final Office Action, paragraph 4. Claims 1, 8 and 9 recite batching tasks of same type, "wherein the tasks of the same type use the same program code."

As discussed above, the Final Office Action does not provide evidence in the record that either Alfieri or Nguyen teach or suggest batching tasks that use the same program code. For at least these reasons, it is respectfully submitted that the rejections of claims 1, 8 and 9 are inappropriate and should be withdrawn.

Claims 2 and 13

Claims 2 and 13 of the pending Application stands rejected as obvious over Alfieri in view of Nguyen. Final Office Action, paragraph 4. The Examiner applies similar reasoning to reject claims 2 and 13. The Applicant therefore uses claim 13 as a representative claim to discuss both claims.

Claim 13 is dependent on and additionally limits claim 11. Claim 13 recites, "executing the task code for processing the new task in the instruction cache without loading new code into the instruction cache."

In rejecting claim 13, the Examiner alleges that Alfieri teaches "on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task." Final Office Action, paragraph 7 and 14. In support of this conclusion, the Office Action cites Alfieri's Abstract and col. 3, lines 29-37. Final Office Action, paragraph 7 and 14.

The Applicant respectfully submits that Alfieri's alleged teachings are not analogous to the claim limitation of "executing the task code for processing the new task in the instruction cache without loading new code into the instruction cache." Claim 13 states that the code for processing a new task is executed without loading the code into an instruction cache, while Alfieri's alleged teaching states otherwise.

Furthermore, the Examiner's conclusions about Alfieri's teachings are not supported by the cited passages.

Alfieri's Abstract states,

> Closely related processing threads within a process in a multiprocessor system are collected into thread groups which are globally scheduled as a group based on the thread group structure's priority and scheduling parameters. The thread group structure maintains collective timeslice and CPU accounting for all threads in the group. Within each thread group, each individual thread has a local scheduling priority for scheduling among the threads in its group. The system utilizes a hierarchy of processing levels and run queues to facilitate affining thread groups with processors or groups of processors when possible. The system will tend to balance out the workload among system processors and will migrate threads groups up and down through processing levels to increase cache hits and overall performance. The system is periodically reset to avoid long term unbalanced operation conditions. Alfieri, Abstract.

Clearly, the Abstract of Alfieri makes no mention or suggestion of executing a loaded code to process the further task when there is a further task of like type in a batch.

The additional citation offered in the Office Action states,

> If a thread is being formed as the initial thread in a new thread group, the new thread group's thread group structure is first created by inheriting the thread group structure of the creating thread's thread group. Then the new thread is created in the newly created thread group. The creating thread assigns the local scheduling and priority to the newly created thread. Unless otherwise specified, the newly created thread will inherit the local scheduling policy and priority of its creating thread. The newly created thread may have a priority that is higher, lower or the same as the priority of the thread that created it. Similarly, individual threads within a thread group may have a priority that is higher, lower or the same as the priority of its thread group.
>
> The thread group is the basic unit of global scheduling across the system. The thread group structure maintains the global scheduling policy and global priority which are used to schedule the execution of the thread group. The thread group structure also maintains the cumulative timeslice and CPU accounting for all threads in its thread group, so timeslicing and CPU accounting records for individual threads within a thread group are not necessary. Each individual thread within the thread group maintains the thread priority and scheduling policy for itself.
>
> The particular method used by a CPU in selecting a thread group to execute is discussed below. Once a particular thread group is selected for execution, the individual thread to be executed is selected based on the local priority and scheduling policy of the threads within the group. Alfieri at col. 3, lines 29-37.

Again, there is no teaching found in this citation of executing a loaded code to process the further task when there is a further task of like type in a batch.

It is respectfully submitted that Alfieri does not teach or suggest executing task code for processing a new task in an instruction cache without loading new code into the instruction cache. Alfieri is concerned with grouping threads accessing and operating on the same data, not program code. Different program code may access the same data, thus it makes no sense for Alfieri to execute one set of program code in place of another just because they both access the same data.

For at least these reasons, and the reasons set forth above for claims 1 and 11, it is respectfully submitted that the rejections of claims 2 and 13 are inappropriate and should be withdrawn.

Claims 3-7, 10, 12, 14 and 16-18

Claims 3-7, 10, 12, 14 and 16-18 are dependent claims and are allowable for at least the same reasons as the independent claims discussed above. Thus, the rejections of claims 3-7, 10, 12, 14 and 16-18 are inappropriate and should be withdrawn.